# Taller Integrador - Python.

### Clase 10

### Sábado 30/05/2020 (Chivilcoy)

### Martes 02/06/2020 CABA.

# Funciones de orden superior.

Repasemos el concepto de función de orden superior.

Si bien Python, es un lenguaje orientado a objetos, permite el desarrollo de aplicaciones o códigos de manera estructurada y funcional, lo cual no es un detalle menor para este lenguaje ya que una de sus principales aplicaciones es la resolución de problemas de ciencia e ingeniería.

Por eso es necesario comprender bien el uso de funciones y ver todas las opciones y herramientas que Python nos pone al alcance de la mano, una de ellas, es la función de orden superior.

Definimos una función de orden superior como aquella capaz de recibir como parámetro a otras funciones y devolver, en casos particulares, una función como resultado.

Esto no es exclusivo de Python, está presente en otros lenguajes de programación y en temas matemáticos.

Resumiendo, y para que quede claro, decimos que una función de orden superior es una función que cumple al menos una de las siguientes condiciones:

1. Recibe una o más funciones como entrada.
2. Devuelve una función como salida.

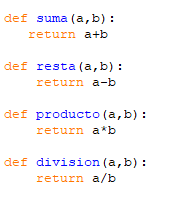
Veamos un ejemplo, resolviendo el ejercicio propuesto del apunte anterior.

Vamos a hacer cuatro funciones, cada una de ellas recibe dos valores como parámetros y realiza una operación matemática.

Luego vamos a escribir una función que reciba como parámetros dos valores y una función.

El objetivo es que esta función, de orden superior, reciba tanto los valores como la función que debe invocar.

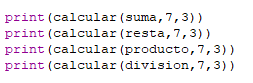
Paso 1: Escribimos las cuatro funciones simples.



Paso 2: Escribimos la función de orden superior.



Paso 3: Probamos el funcionamiento invocando la función de orden superior para que, a su vez, invoque a las cuatro funciones simples.



Este es un ejemplo sencillo, y vemos a simple vista que sería mas sencillo invocar por separado cada una de las funciones que invocar la de orden superior.

Pero esta posibilidad de pasar una función como argumento para otra, es aprovechado por Python para incluir algunas herramientas mas que podemos utilizar en la programación funcional.

### Función FILTER.

Esta función toma como entrada, una función y una lista de elementos sobre los cuales aplicará dicha función.

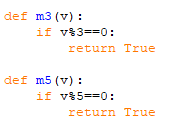
Es decir, utiliza todos los elementos de la lista como argumento para la función que pasamos como argumento, y nos devuelve una lista filtrada solo con los elementos que satisfacen a la función simple pasada como argumento.

Veamos un ejemplo sencillo.

Vamos a escribir dos funciones simples, una que determine si un valor es múltiplo de 3 y otra que determine si un valor es múltiplo de 5.

Luego vamos crear una lista con números y, mediante el uso de la función FILTER, vamos a extraer de dicha lista, primero los múltiplos de 3 y luego los múltiplos de 5.

Paso 1: escribimos las funciones simples.



Paso 2: creamos una lista de números.



Paso 3: Utilizamos FILTER con ambas funciones.



Y este es el resultado de aplicar nuestros filtros a la lista de números.



Este es un ejemplo sencillo, pero imaginemos las posibilidades de aplicar funciones tan complejas como necesitemos a una lista muy grande de valores que, como ya vimos en otras clases, podemos obtener directamente de bases de datos o archivos.

### Funciones LAMBDA.

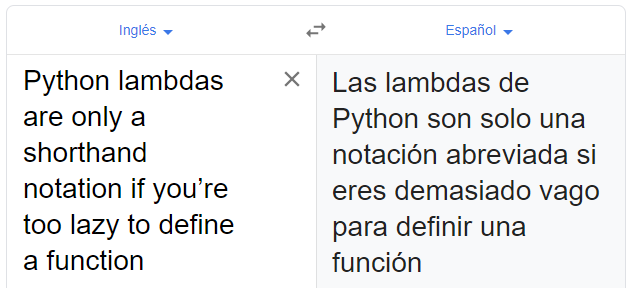
La mejor definición de funciones LAMBA la podemos sacar de la documentación oficial de Python:

<https://docs.python.org/3/faq/design.html#why-can-t-lambda-expressions-contain-statements>

Donde encontramos textualmente lo que sigue:

“Python lambdas are only a shorthand notation if you’re too lazy to define a function.”

Para los que no andamos bien con el inglés, podemos hacer uso del traductor de Google:

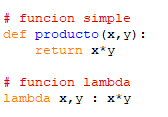


Y es justamente eso, una forma abreviada de definir funciones sin necesidad de ponerles un nombre, por lo que comúnmente leemos en algunos textos que se trata de funciones anónimas.

La sintaxis es muy sencilla, utilizando la palabra reservada LAMBDA seguida de los parámetros, dos puntos y a continuación la condición.

LAMBDA parámetro1, parámetro2: condición u operación.

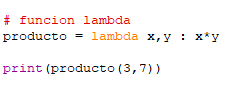
Veamos un ejemplo sencillo, una función LAMBA que nos devuelva el producto de dos números.



La primera es la definición tradicional que conocemos, en la que asignamos a la función un nombre.

La segunda es la función LAMBDA, ambas hacen exactamente lo mismo.

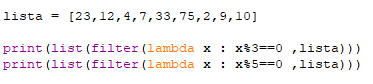
Para utilizar una función de estas, tenemos que asignársela a una variable, por ejemplo:



En definitiva, estaríamos hacienda casi lo mismo que si hubiésemos definido la función de manera tradicional.

El objetivo de estas funciones anónimas, es poder simplificar la escritura cuando trabajamos con funciones de orden superior, por ejemplo, la recientemente vista función FILTER.

Probemos modificar el código reemplazando las funciones simples con funciones LAMBDA en el ejemplo de FILTER con múltiplos de 3 y de 5.



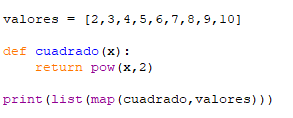
Si ejecutamos el código, obtendremos el mismo resultado que obtuvimos cuando habíamos definido las funciones simples, de la manera tradicional asignándoles un nombre.

### Función MAP.

Esta función es de uso similar a la función FILTER con la diferencia que la función simple recibida como parámetro, se aplica a todos los elementos de la lista que se pasa junto con la función y el resultado es una lista con la misma cantidad de elementos que la lista original.

Veamos un ejemplo sencillo, tenemos una lista con números y una función que recibe un valor y lo eleva al cuadrado.

Mediante el uso de MAP aplicamos la función a todos los elementos de la lista.



Y este es el resultado que vemos pantalla:

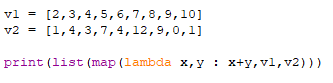


Podemos modificar el código del ejemplo para utilizar una función LAMBDA:



Uno de los beneficios de la función MAP es que evita el uso de estructuras de repetición.

Además, podemos utilizarlo sobre mas de una lista u objeto iterable, por ejemplo, tenemos dos listas de números y queremos obtener la suma de los elementos que tengan el mismo índice:



La diferencia con la sintaxis que veníamos usando, es que ahora estamos usando dos listas (y podemos usar más) como tercer argumento para la función MAP.

Resultado:

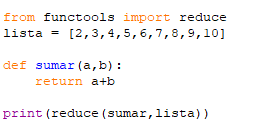


### Función REDUCE.

Esta es otra función de orden superior, toma como argumentos una colección de datos y los reduce a un solo valor en base a una función simple que también le pasamos como argumento.

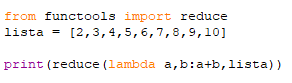
Esta función se encuentra dentro del módulo FUNCTOOLS, por lo que deberemos importarla.

Por ejemplo, queremos sacar la sumatoria de todos los valores cargados en una lista.



Lo que hace el código es sumar todos los valores de la lista, REDUCE va haciendo el acumulado de cada suma antes de seguir con el siguiente elemento de la lista.

También podemos hacer uso de una función LAMBDA:



Básicamente lo que hace REDUCE es hacer un cálculo acumulativo sobre una serie de valores y devolver el resultado.